State-Per-Transfer Technology

An Application Note

www.arium.com

Introduction:

In the past, with traditional microprocessors, bus transactions were very orderly and sequential. This made it easy to capture bus activity and trigger on events using a general purpose logic analyzer. As microprocessors have evolved, the simple bus protocol has given way to performance enhancing techniques involving pipelining, multilevel caching, and overlapping transactions. These enhancements have also made it more difficult to capture bus transactions and trigger on real-time bus events. Intel's Pentium® Pro and Pentium II processors are perfect examples of processors that have used these complex techniques to gain enhanced performance.

Responding to the needs of system designers and software developers, American Arium has adopted a technology that makes it possible to easily capture execution history and trigger on complex bus transactions in real-time. The industry term for the technology is state-per-transfer.

The Traditional Microprocessor Bus

As mentioned previously, traditional microprocessor bus transactions were very orderly and sequential, making them easy to work with. The example in Figure 1 depicts the bus transactions that one might expect from a traditional processor. Note the simple linear fashion in which instructions are fetched except when a load, store, or branch is encountered. Virtually any logic analyzer could capture these transactions and trigger on specific events.

Address	Control	Instruction/Data
8D000F00	101	Instruction Fetch (ADD)
8D000F04	101	Instruction Fetch (STORE)
20000CE0	011	Memory Data Store
8D000F08	101	Instruction Fetch (LOAD)
20000C80	111	Memory Data Load
8D000F0C	101	Instruction Fetch (JUMP)
9FF10008	101	Instruction Fetch (AND)

Figure 1: Traditional microprocessor bus operation

The Pentium Processor Bus

With the introduction of more advanced processors such as the Pentium processor, things became a bit more complicated. Bus transactions are less sequential and more complex. In addition, processor activity is sometimes not visible on the processor bus because data and instructions might be accessed from the internal cache. This situation forces some hardware debuggers to disable cache in order to trace processor activity. The problem with disabling cache is that timing and sequence are altered which often conceals the fault. American Arium's ICE products with trace have the ability to display all code execution with cache enabled. We do this by using Branch Trace Message (BTM) cycles.

A BTM is a special cycle that Pentium processors can present on the bus whenever a branch is made in the execution flow. Each BTM describes the branch being taken and allows sophisticated debugging tools such as those made by American Arium to trace execution of cached instructions. Figure 2 shows a code fragment that is executed entirely out of cache. As this code executed, one section looped a total of five times. Since the code was executed out of cache, BTM cycles were the only events that occurred on the bus. American Arium products with trace have the ability to capture these BTM cycles in real-time and then "connect the dots" to show the entire execution history.

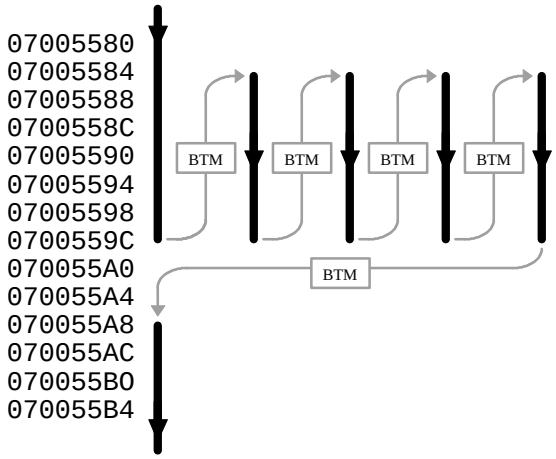

Figure 2: Execution of cached instructions showing BTMs

Figure 3 is a trace display that resulted from a Pentium processor executing code entirely out of cache. Note the special bus cycles identified as "SPBUS" which were the only bus cycles that occurred. We can identify them as BTM cycles because they are special bus cycles with the byte enables (BE) encoded as "DF" (see the Intel manuals for a full description). The disassembly lines between the BTM cycles do not have sequential state numbers because they did not occur on the bus. They were determined through analysis of the BTM cycles by an American Arium ICE.

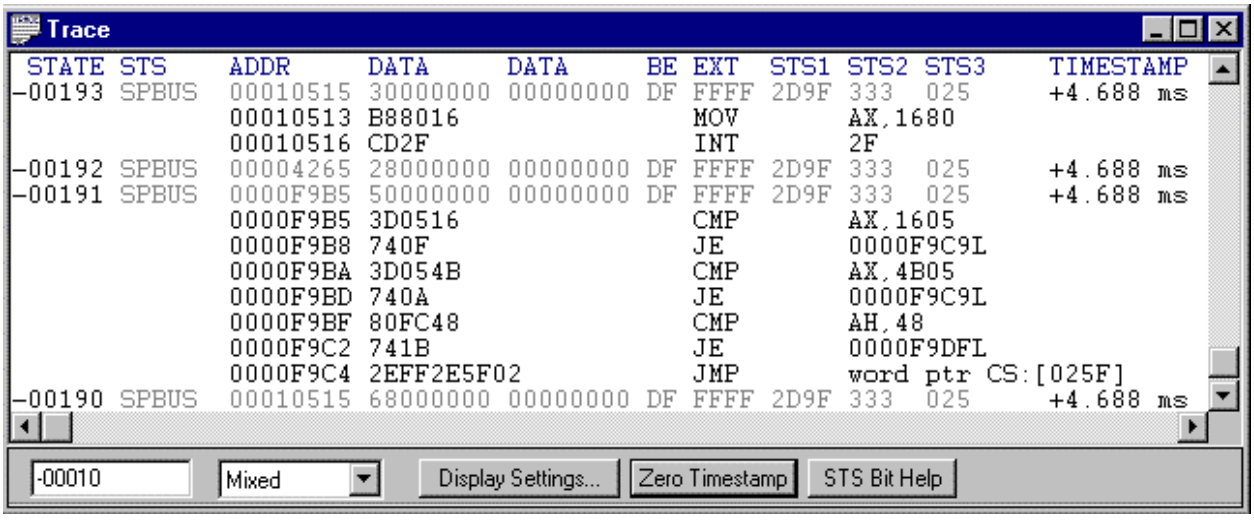

Figure 3: American Arium trace window

From the previous example, it is easy to see how difficult it would be to capture and trigger on these events with a standard logic analyzer since only BTM cycles were present the bus. Things become even more complicated with the P6 bus protocol used by the Pentium Pro and Pentium II processors.

The P6 Bus

Bus operation made a significant leap in performance and complexity with the introduction of the P6 bus. The P6 bus protocol is used by Pentium Pro and Pentium II processors. Earlier Pentium® processors worked well in a single processor environment but could not provide efficient bus utilization when multiple processors were involved. The P6 bus was enhanced with the following features:

- Direct support of as many as four processors clustered on a single bus.
- Slow response agents can go off-line and complete the transaction later (deferred reply).
- The ability to have as many as eight overlapping bus transactions.

All of these features complicate debugging, but the overlapping of bus transactions creates the greatest challenge in trying to capture and trigger on bus events in real-time. The original Pentium processor had the ability to overlap internal instruction execution but now the P6 bus can also overlap up to eight bus transactions. Each transaction is composed of as many as five different phases and all of these phases may occur at the same time with each phase tied to a separate transaction. This is illustrated in Figure 4, which shows eight overlapping bus transactions

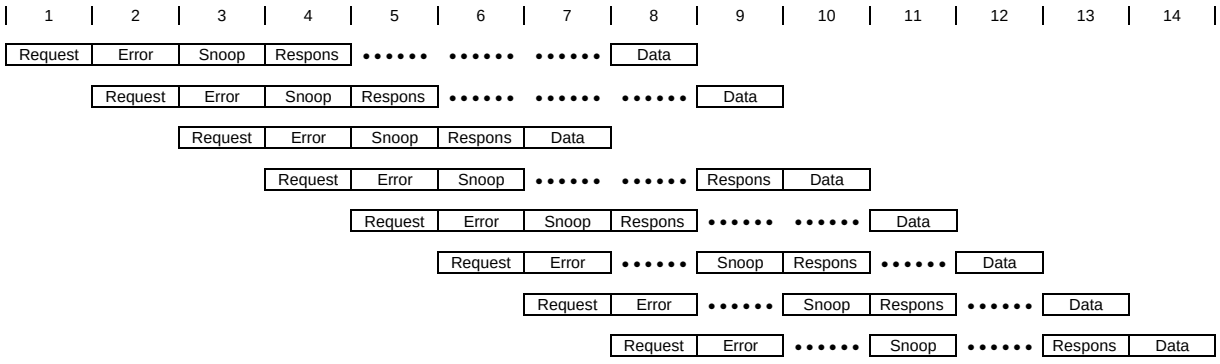

Figure 4: P6 overlapping bus transactions

The P6 Debugging Challenge

How does this bus protocol affect your efforts to debug firmware or software? It affects you a great deal if you need a tool that can break on a bus event such as writing a specific value to a particular memory location. Suppose that you wanted to add a breakpoint such as the one shown below in Figure 5. This window shows that we are planning to break on the first occurrence of the value FFE0 being written to location 00004809. Most in-circuit emulators and logic analyzers cannot provide you with this kind of real-time breakpoint based on bus activity.

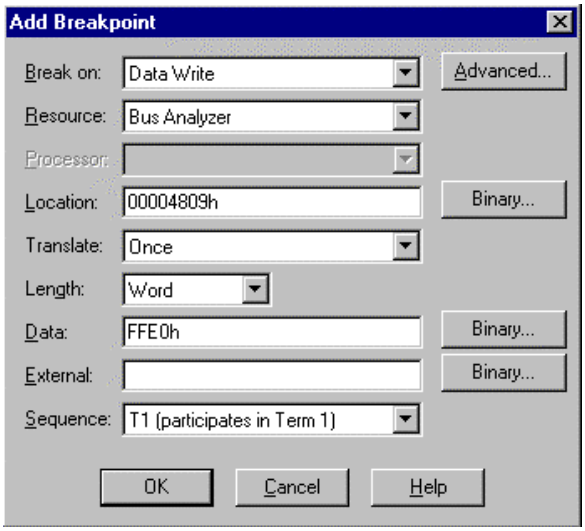

Figure 5: American Arium breakpoint window

Figure 6 helps explain why this is so difficult for most in-circuit emulators to accomplish in real-time. Suppose we wish to trigger on what we will call Transaction A. This transaction is spread across a total of seven bus cycles. In order to trigger on this transaction, an in-circuit emulator must capture each phase of every cycle, store them, correlate them to one of the eight transactions that may be in progress, and test for a trigger condition. With the bus operating at 100 MHz or more, this must be all be done in just a few nanoseconds. Software cannot do this in real-time. It can only be done with complex hardware.

Bus Cycle	Request A	Error Z	Snoop 2	Response X	Data W
Bus Cycle	Request B	Error A	Snoop 3	Response Y	• • • • • •
Bus Cycle	Request C	Error B	Snoop A	Response Z	• • • • • •
Bus Cycle	Request D	Error C	Snoop B	• • • • • •	Data X
Bus Cycle	Request E	Error D	Snoop B	Response A	Data Y
Bus Cycle	Request F	Error E	Snoop C	Response B	Data Z
Bus Cycle	Request G	Error F	Snoop D	Response C	Data A

Figure 6: A single P6 bus transaction spread over seven bus cycles

American Arium uses industry defined state-per-transfer technology to accomplish this feat. It consists of a sophisticated high speed logic array that enables our products to not only capture P6 bus activity but also trigger on a wide range of user defined events that can be of a singular or sequential nature. Figure 7 shows a block diagram of one of our typical ICE products with trace.

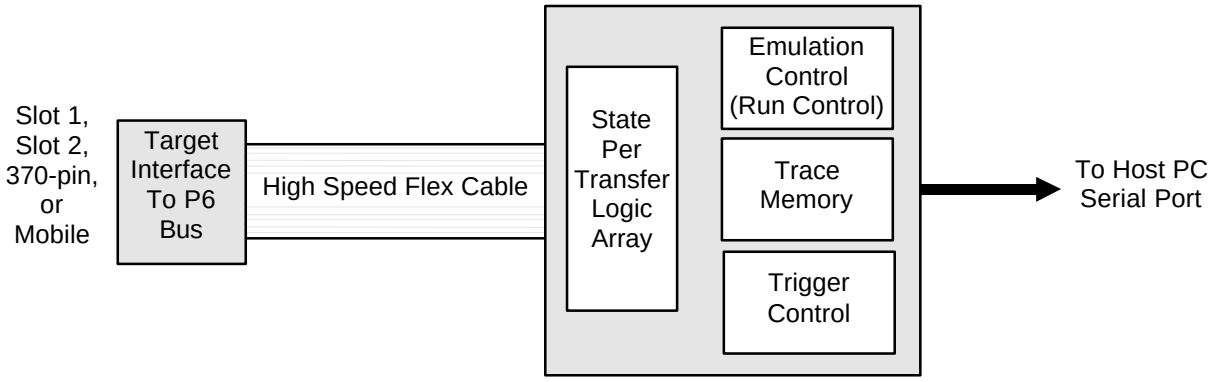

Figure 7: Block diagram of American Arium TRC-P6, TRC-S1, TRC-S2, TRC-SA, and TRC-MM

With state-per-transfer technology, precise multilevel breakpoints can be realized. This technology also allows us to produce a trace display of execution history with all phases of each bus transaction aligned into a single state as shown in figure 8. Only American Arium uses this technology that provides designers and developers with the tools needed to effectively debug target systems and meet project deadlines.

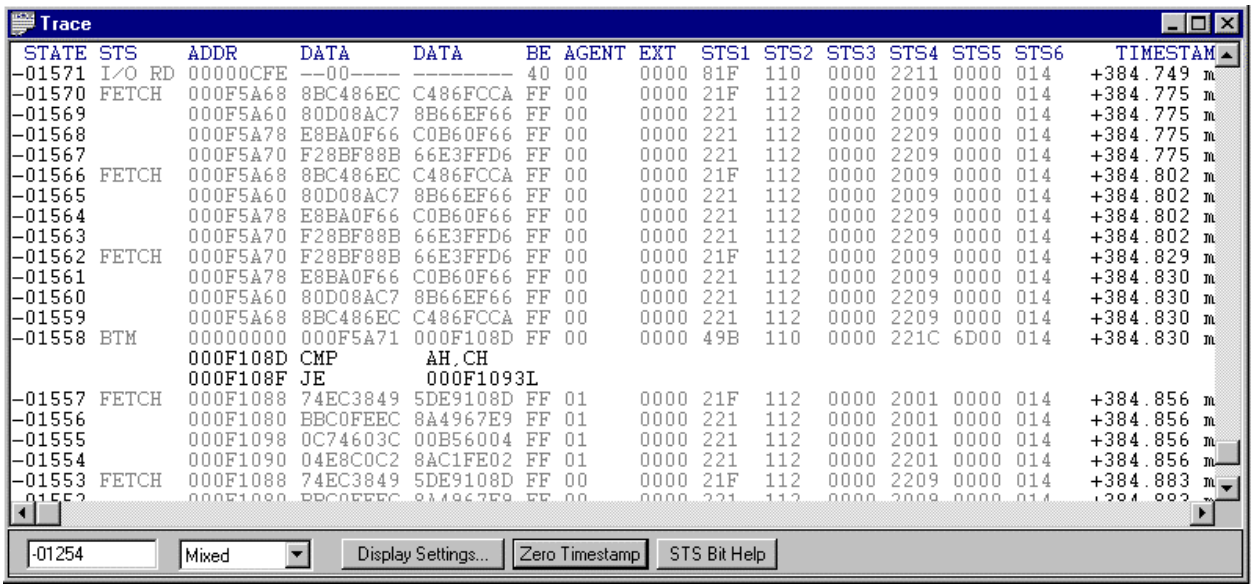

Figure 8: American Arium trace window showing an aligned trace display from a P6 bus

References:

- Pentium Pro Processor System Architecture, by Tom Shanley, Mindshare Inc., published by Addison Wesley Developers Press, 1997, ISBN: 0-201-47953-2
- Pentium Pro Processor Developer's Manual, Volume 1: Specifications, published by Intel Corporation, 1995, 1996, 1997, Intel Order Number 242690

14281 Chambers Road
Tustin, CA 92780
Voice: 714-731-1661
Fax: 714-731-6344
Web: www.arium.com
E-mail: info@arium.com